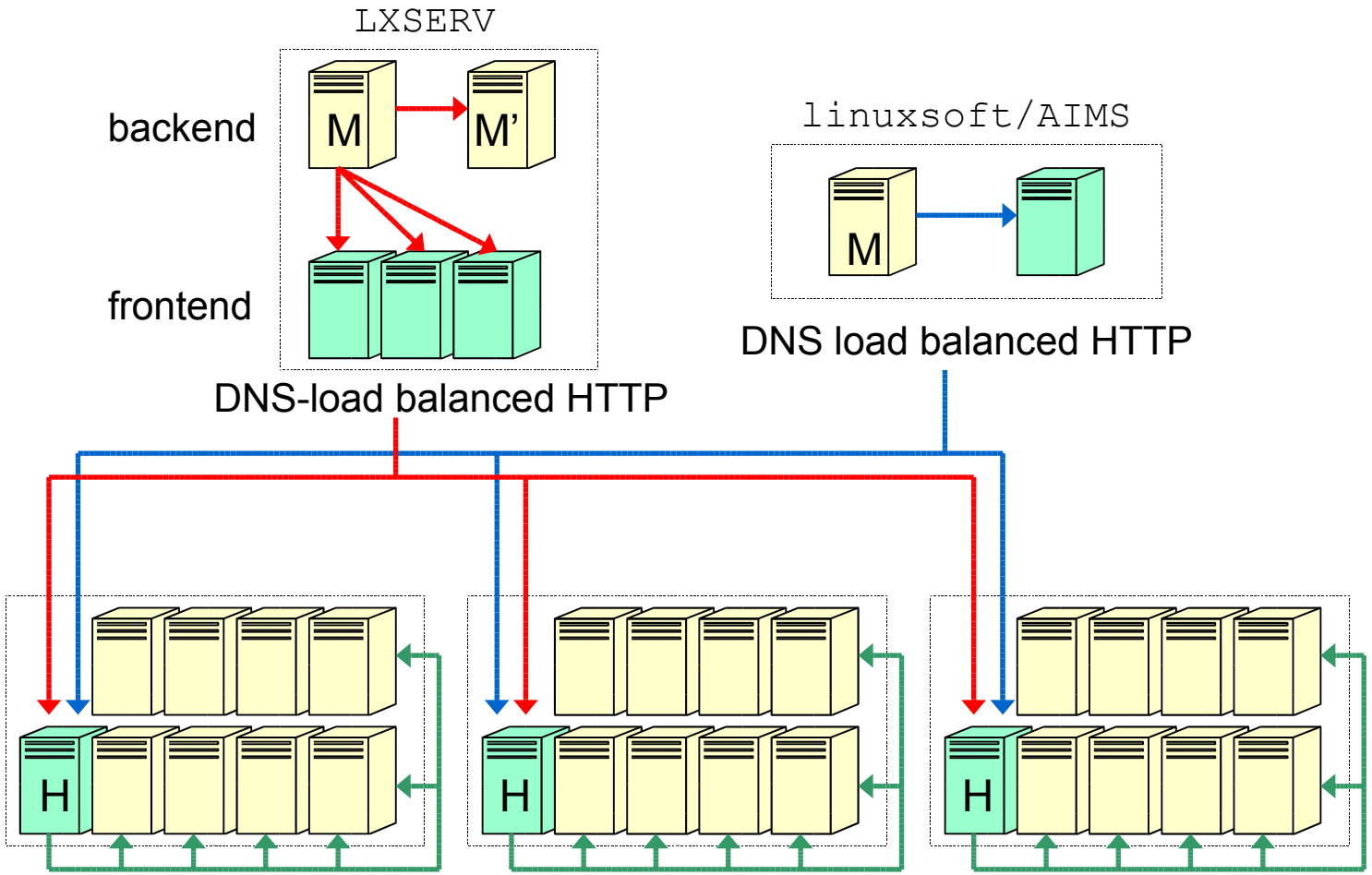
# *Proxy servers revisited*

Marco Emilio Poleggi
Marco.Poleggi@cern.ch
12/04/2005

# Outline

- Proxy architecture
- Reverse proxies
- Apache 2.0 migration
- Squid vs Apache
- Proposal: two-tier caching
- New NCM component: *ncm-rproxy*
- Deployment on CERN-CC clusters

# Proxy architecture




- See "Proxy servers in CERN-CC", Germàn Cancio, 02/03/04, `http://agenda.cern.ch/fullAgenda.php?ida=a04930`
  - Basic concept
  - Current deployment

# Proxy architecture details

- Two-tier proxy-caching hierarchy:
  - Cluster-side caching: one *head-node* per cluster decouples cluster nodes from the server tier
    - Cluster nodes (clients) talk to their head-node, as if it were the origin server
    - Head-nodes forward requests to the server tier
  - Server-side caching: many DNS-load balanced front-end proxies decouple clusters from the back-end server
    - Unique DNS name for front-ends
    - Front-ends forward requests to the back-end server
    - The back-end is a standard HTTP server
- Based on *reverse* proxies
  - *Apache* (originally rel. 1.3, now 2.0?) or *Squid* servers
  - Semi-transparent to clients

# Reverse proxies

- They act as “dispatchers” towards (possibly) different repositories according to a given mapping
  - Content-based mapping: *path* ↔ *URL*
    - `/swrep` ↔ `http://lxservb01/swrep`
- Requested objects can be locally cached
  - Memory cache: very popular small objects
  - Disk cache: less popular big objects
  - Cacheable objects:
    - static/long-lived: software packages (RPM’s, PKG's, ...)
    - dynamic/short-lived: XML profiles, ...
  - Uncacheable objects: those generated on-the-fly (CGI/ASP/JSP results)

# Apache 2.0 migration

- Configuration file can be split
  - More control over module directives: renaming/removing a conf file disables the module
  - Proxy/SSL/... configuration in separate files
- Cache support is now modular inside `mod_proxy`
  - `mod_cache + mod_mem_cache + mod_disk_cache`
  - More flexible
    - Different caching strategies can be adopted according to pathnames and sizes: selective caching in *main memory* or on *disk* (or both)
  - More complex configuration
    - Some parts may depend on other control directives
    - Not always possible to arbitrarily append missing directives
- Uniform logging directives

It seems fine, but...

# Apache 2.0 migration (II)

- Cache porting incomplete! As of rel. 2.0.46 (SLC3.0.4):
  - Memory caching not fully reliable
    - Apparently, HIT after many consecutive MISSes
  - No garbage collection for disk caching!
    - Not even for the latest 2.0.53 rel.
    - Helper program *htcacheclean* from Apache 2.1 (alpha) can be used, but requires a local build (not distributed as package)
  - Logging of caching information is fuzzy
    - Statistic analysis on log files not possible
    - Maybe a race condition is fixed in rel. 2.0.53
- What to do?
  - Fall-back to Apache 1.3 is problematic in SLC3...
  - Test Apache 2.0.46 + disk-caching + htcacheclean
  - Test Apache 2.0.53 + mem/disk-caching + htcacheclean
  - Try *Squid*

# Squid vs Apache

- Reverse proxy in *accelerator* mode via a translation layer
- Caching-proxy only
  - Must use also Apache if Web server needed
- Advanced cache management: hierarchies, ICP, ...
  - Cooperative caching could be interesting
  - Native in-memory caching of "hot" objects
  - Cache statistics via CGI (Web server required)
- Recompilation might be required
  - Disable/enable some default options
  - Patch for custom logging a la Apache
    - → Statistic analysis through dedicated tools such as *Webalizer*
- Configuration not straightforward for multiple back-ends
  - External *redirector* helper needed

## *If things go well…*

# Proposal: a two-tier caching strategy

- Both memory and disk are used
  - "small" objects in memory and "big" ones on disk
    - Separated caches: no room wasted
    - mem-cache should settle to holding the "working-set", i.e., the set of most popular files
  - Try to cache in memory first
  - Fast access to popular files: should bear traffic surges during large updates/upgrades
  - Plenty of disk space for large files
    - To avoid engaging back-ends and network with long transfers
- Main tuning knobs (to maximize the hit ratio)
  - Cache sizes
  - Access size threshold between memory and disk

# Proposal: a two-tier caching strategy (II)

- Why? For *Web* objects:
  - File popularity (frequency of occurrence of the *r-th* ranked item) is *Zipf*-like: $P(r) \sim r^{-b}$, $b \sim 1$
  - File-size distribution is *heavy-tailed*: $P[X>x] \sim x^{-a}$, $x \to \infty$, $0<a<2$
  - Correlation: small files are more popular
    - Working-set much smaller than file-set
    - Popular files are in the distribution's body, less popular ones lie in the tail
- But, what about *package/XML* objects?
  - Probably similar distribution... it could be verified, since we know the file-set

# *New NCM component:* ncm-rproxy

- *ncm-rproxy* configures a standard Apache 2.0 as a reverse proxy-caching server
  - Support for both disk-cache and mem-cache... for a possibly nice future ;-)
  - Most important configuration directives supported
  - Minimal effort re-configuration:
    - Modifies a main configuration template then merges it with the standard Apache's configuration file
    - Modifies a proxy-dedicated configuration template then copies it inside the Apache's configuration tree
  - All changes are initially made to temporary files, then committed, if needed

# *New NCM component:* ncm-rproxy *(II)*

- Three-tier configuration:
  - `/software/components/rproxy/httpd/`: basic Apache's directives
    - mem/disk-cache enabling
  - `/software/components/rproxy/proxy/`: proxy-related directives
    - Restricted access from a given domain
    - Definition of *path* ↔ *URL* mappings
  - `/software/components/rproxy/cache/`: cache-related directives
    - Common options: expire times, ...
    - mem-cache options: cache size, maximum cacheable object size
    - Disk-cache options: cache root and size, minimum/maximum cacheable object size

# Deployment on CERN-CC clusters

- Currently deployed on *LXPLUS/SLC3*
  - Head-nodes: *lxc1m990* and *lxc1m991*
  - Apache 2.0.46 + disk-cache [+ mem-cache]
- Configuration
  - Default/maximum expire time: 1 day
  - Mem-cache size: 100MB
  - Mem/disk-cache size threshold: 1MB
  - Disk-cache size: 6GB
    - → No garbage collection! However, the file-set is bounded
  - Max disk-cache object size: 100MB
- Performance study
  - Need benchmarking and/or statistic analysis of logs...
    - → Dedicated tools, such as *Webalizer*
  - *lemonweb* for overall behaviour